A Dual Back-Propagation Scheme for Scalar Reward Learning

Paul Munro
Interdisciplinary Department of Information Science
University of Pittsburgh
Pittsburgh, PA 15260

Abstract. Explicit supervised learning rules [e.g. the delta rule] require that each of the output units in a network receive a training signal indicating the "correct" response value; the unit can then adjust its parameters, so that its future response to the same stimulus is closer to the desired value. A much more realistic assumption for the nature of a supervisory signal is a single scalar "goodness-of-response" or "reward" signal. This *credit assignment problem* is handled here by a supervisory network which monitors the activities of both the sensory and effector units, and learns to predict the value of the reward signal using the generalized delta rule of Rumelhart, Hinton, and Williams (1986). The activity of a particular "predictor unit" thus comes to be associated with the expected reward. Having learned to mimic the environment's reward criteria, the supervisory network can provide each effector unit, by way of a back-propagation scheme, with an individualized correction signal that will lead to increased activity in the predictor. The actual reward is hence enhanced to the extent that the predicted reward is reliable.

The Problem

One of the most attractive features of the Parallel Distributed Processing approach to understanding cognition is its inherent framework for adaptation. Learning is implemented by mechanisms that make small adjustments to the weights such that over time the response becomes more appropriate in some sense. A particular mapping from a set of input patterns to a set of output patterns can be "programmed" in to a network using the so-called back-propagation algorithm, or generalized delta rule (Rumelhart, Hinton, and Williams, 1986). The formidable power of this modification rule as a model for learning in real-world situations is somewhat offset by the need for a "supervisor"; that is, this rule requires that learning can only take place when *each output unit in the network is given detailed error information.* This is a general problem of supervised learning rules, and various attempts have been made to exploit the power of

such rules and to eliminate the need for a supervisory signal. [A notable example is the encoder network (Ackley, Hinton, and Sejnowski, 1985), which is taught to replicate its input pattern as its output; hence the input pattern provides the detailed pattern of the desired output.]

While feedback from the environment may be inadequate with regard to its detail, feedback of some kind is essential for effective learning. In order to learn, the response of an organism must be evaluated at some level. This evaluation need not be generated by a teacher *per se;* for example, if a young animal swallows food, the reward takes the form of hunger satisfaction. The problem of assigning proper credit (or blame) to the individual output units that generated the evaluative feedback is known as the *credit assignment problem.*

The Network Configuration

The network described in the present paper is built in two stages. A low-level network (henceforth LOWNET) maps the sensory input to the organism into an effective response, that is, into a response that has an effect on the external world [e.g. a "motor response"]. Another network (HIGHNET) monitors the activities of both the sensory and effector units. The probability of reward contingent on response is assumed to depend solely upon instantaneous environmental information which is *perceivable,* i.e. the sensory pattern contains information sufficient to determine a "good" response. HIGHNET has access, therefore, to all the information necessary to mimic the reward signal and hence, one of the units in HIGHNET receives reward information as a training signal, computes an errror signal, and propagates it backwards through HIGHNET using the back-propagation procedure.

Note that for a *given configuration of weights* in the network, the entire pattern of activation throughout *both* LOWNET and HIGHNET is determined by the environment through its activation of the sensory units [the environment can influence the *modification* of the weights by way of the reward signal, but the reward has no direct influence on the response of the network]. As HIGHNET learns to mimic/predict the reward signal, LOWNET begins to adapt such that the net input, consisting of the

combined pattern of sensory and motor activity, to HIGHNET will maximize the *mimicked* reward signal, for a given configuration of weights in HIGHNET. To accomplish this, HIGHNET translates the scalar reward signal into a more detailed pattern of error information for the motor units.

HIGHNET thus serves as a "mental model" of the environment, in that it must learn to accurately predict combinations of sensory-motor activity that yield favorable feedback from the environment.

The full network architecture, incorporating LOWNET and HIGHNET, is displayed in Figure 1. It consists of five distinct populations of units: a set **S** of sensory units is activated by some physical stimulus; a set **K** receives activation from **S**; a set **M** of motor units receives activation from **K** [the population **M** directly controls the organism's motor response]; a set **H** receives activation from both **S** and **M**; and set **R** [which consists of but a single unit] receives activation from **H**.

Thus, the network can be viewed as two overlapping three-layer feed-forward networks, each made up of an input layer, an intermediate (hidden unit) layer, and an output layer. Each of the two networks is strictly layered; that is, the hidden units receive activation from the input units only and the output units receive activation from the hidden units only. The five population sets described above map onto LOWNET and HIGHNET as outlined in Table 1. The architecture can be generalized to arbitrary numbers of intermediate layers in either network, and the input to HIGHNET may presumably be expanded to include units from the intermediate layer(s) of LOWNET.

TABLE 1. Layered Structure of LOWNET and HIGHNET

NETWORK	LAYERS		
	Input	*Intermediate*	*Output*
LOWNET	**S**	**K**	**M**
HIGHNET	S∪M	**H**	**R**

Network Processing

Notation. The activity levels of the various units throughout the network are denoted by the lower case letter corresponding to the population, with a subscript specifying the individual unit (see Figure 1); for example, s_2 is the activity level of unit 2 in the population set **S**. The single unit occupying population **R** is an exception; its activity level is denoted r (with no subscript). The reward, or evaluative signal, from the environment is denoted e. The weight values are specified by the symbol w with a superscript denoting the "postsynaptic" population set *followed by* a superscript denoting the "postsynaptic" population set, and subscripts denoting the individual units within those sets.

All the units in the network, with the exception of those receiving direct input from the environment, compute their responses according to a semi-linear rule, that passes a weighted linear sum of the inputs through a nonlinear nondecreasing continuous function (f). Hence, a generic unit computes its activity y_i as a function of its input values x_j and the corresponding weights w_{ij} according to the formula

$$y_i = f\left[\sum_j w_{ij}\, x_j\right] \qquad [1]$$

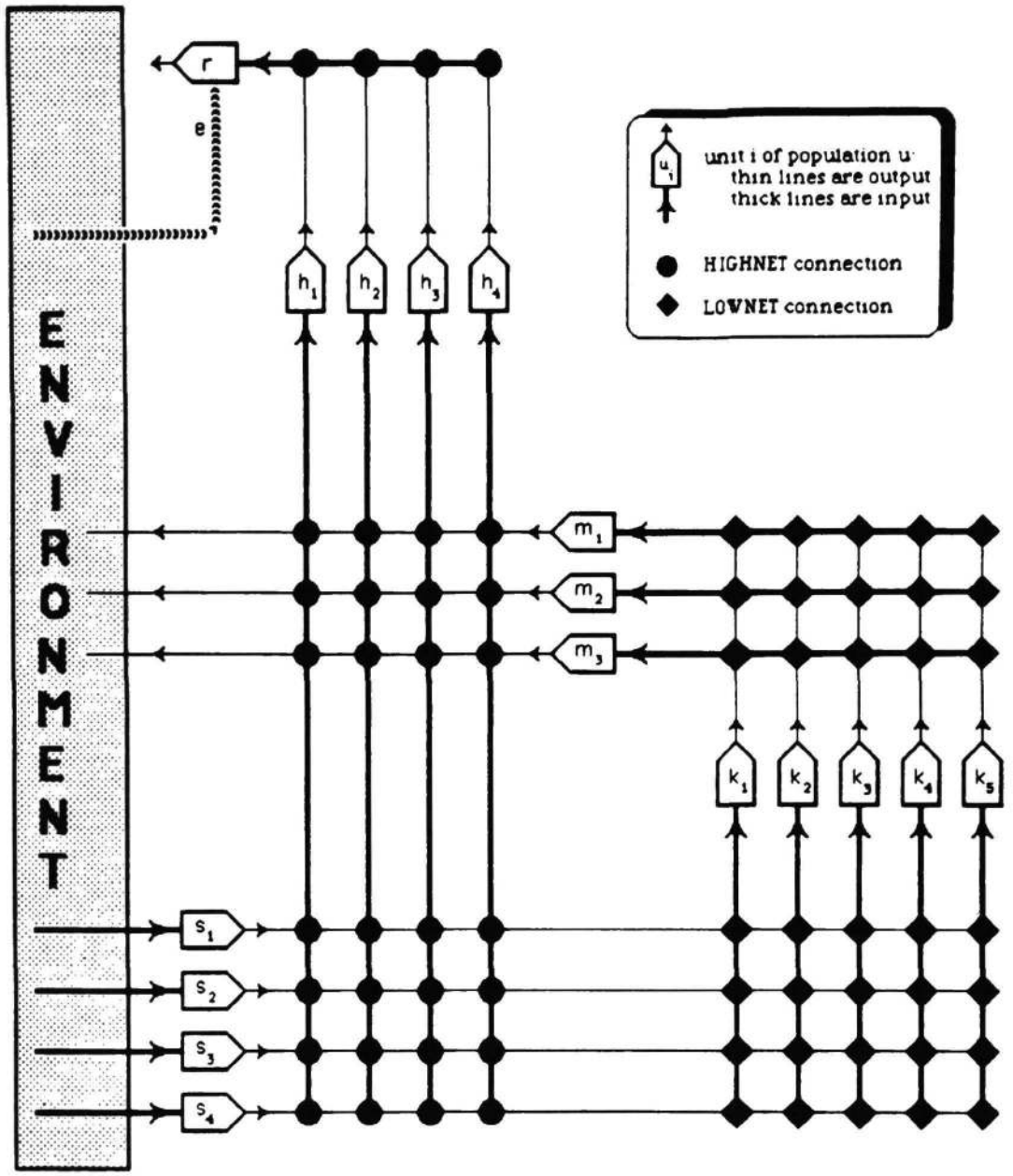


Figure 1. *Architecture.* An example network is illustrated. Sensory input across the **S** population activates the **K** population and in turn the motor population (**M**) via the connections of LOWNET [◆]. The combined activities of the sensory and motor populations provide the input to HIGHNET. The connections of HIGHNET [●] are modified such that the unit r comes to predict the reward signal e.

Applying (1) to the network architecture described above yields the following set of equations for the activity values:

$$k_i = f\left[\sum_j w_{ij}^{KS} s_j\right] \qquad [2a]$$

$$m_i = f\left[\sum_j w_{ij}^{MK} k_j\right] \qquad [2b]$$

$$h_i = f\left[\sum_j w_{ij}^{HS} s_j + \sum_j w_{ij}^{HM} m_j\right] \qquad [2c]$$

$$r = f\left[\sum_j w_j^{RH} h_j\right] \qquad [2d]$$

Two Concurrent Learning Procedures

Rationale. A learning algorithm for minimizing a quantity Q can be obtained by the gradient descent assumption

$$\Delta w_{ij} = -\frac{\partial Q}{\partial w_{ij}} \qquad [3]$$

Removal of the minus sign from [3] will result in a learning rule that seeks *maximum* values of Q by gradient *ascent*.

If the dependence of the reward signal on the weights of LOWNET were known, the gradient descent technique could be applied directly to LOWNET, and there would be no need to introduce any more complexity to the network. However, the requisite partial derivatives depend on details of the environment that are not known to the naive organism. However, a two-stage application of gradient descent rules can be used to get at this problem. Taking both the sensory and the motor activities as input, HIGHNET generates a *predicted reward signal* r. The connections of HIGHNET are modified so as to perform a gradient descent in the squared difference between r and the actual reward e.

Since the dependence of r (but not e) on the connection strengths of LOWNET can be computed in terms of network variables; hence they follow a different learning rule, one which performs gradient ascent in r. Thus the learning dynamics are based on the following assumptions:

$$\Delta w_{ij} = -\frac{\partial (e-r)^2}{\partial w_{ij}} \qquad [w_{ij} \in \text{HIGHNET}] \qquad [4a]$$

$$\Delta w_{ij} = \frac{\partial r}{\partial w_{ij}} \qquad [w_{ij} \in \text{LOWNET}] \qquad [4b]$$

From these assumtions, learning rules are to be derived for both nets that are "factorable" in the sense that the expression for the change in any given weight is the product of a postsynaptic factor and the presynaptic activity value. For HIGHNET, the postsynaptic factor is referred to as the **effective predictor error** δ and for LOWNET it is the **effective predictor enhancement** ε; since these values depend on the response at the top level (**R**) and are propagated back down the network, the ε value must be evaluated for (or by) all units in the network, whereas computation of d is required for the HIGHNET units (sets **R** and **H**) only. The values of δ and ε for each of the population sets and the modification rules for their input connectivities are given in Table 2.

TABLE 2. Back-Propagation Formulae for the Entire Network

Network	*Population*	*Eff. Error*	*Eff. Enhancement*	*Modification Function*
HIGHNET	**R**	$\delta^R = (e-r)\ f'(r)$	$\varepsilon^R = f'(r)$	$\Delta w_i^{RH} = \delta^R h_i$
	H	$\delta_i^H = w_i^{RH} \delta^R f'(h_i)$	$\varepsilon_i^H = w_i^{RH} \varepsilon^R f'(h_i)$	$\Delta w_{ij}^{HX} = \delta_i^H x_j$ [where $\mathbf{X} \in \{\mathbf{M},\mathbf{S}\}$]
LOWNET	**M**		$\varepsilon_i^M = \sum_j w_{ij}^{HM} \varepsilon_j^H\ f'(m_i)$	$\Delta w_{ij}^{MK} = \varepsilon_i^M k_j$
	K		$\varepsilon_i^K = \sum_j w_{ij}^{MK} \varepsilon_j^M\ f'(k_i)$	$\Delta w_{ij}^{KS} = \varepsilon_i^K s_j$

Simulation Methodology

The environmental evaluation (reward) function was evaluated as follows. Each sensory pattern was mapped onto a unique motor pattern. Deviations from this response resulted in a diminished evaluation signal e, which was computed as the negative exponential of the distance from the motor response vector **m** to a "target" response **t**.

$$e = \exp\left[-\sqrt{(\mathbf{t}-\mathbf{m})\cdot(\mathbf{t}-\mathbf{m})} \right] \qquad [5]$$

In each case, the two kinds of learning were performed *separately*. HIGHNET would learn first (for several hundred thousand pattern presentations) by generating *random* motor responses for each sensory stimulus presentation. This **flailing strategy** allowed HIGHNET to explore the shape of the evaluation function over the space of possible sensory-motor combinations. It was found that learning was too slow if the random variables for the motor units were distributed uniformly over the range of the possible response values. Hence, during flailing, the units in set **M** took on either the minimum or maximum values of the function f [in this case, $\pm$ 1].

Once HIGHNET seemed to have identified the peak of the reward function for all, or nearly all, of the sensory patterns, the modification of HIGHNET connections ceased and LOWNET was permitted to function, both as the determiner of the output (no more flailing), and as a system of dynamic connection strengths. The learning in LOWNET then proceeds to drive up the value of the estimated reward, r.

Using this learning procedure, the network was guided to associate two paired pattern sets ($\mathbf{s} \rightarrow \mathbf{t}$) on the basis of a reward signal alone.

Simulation Results

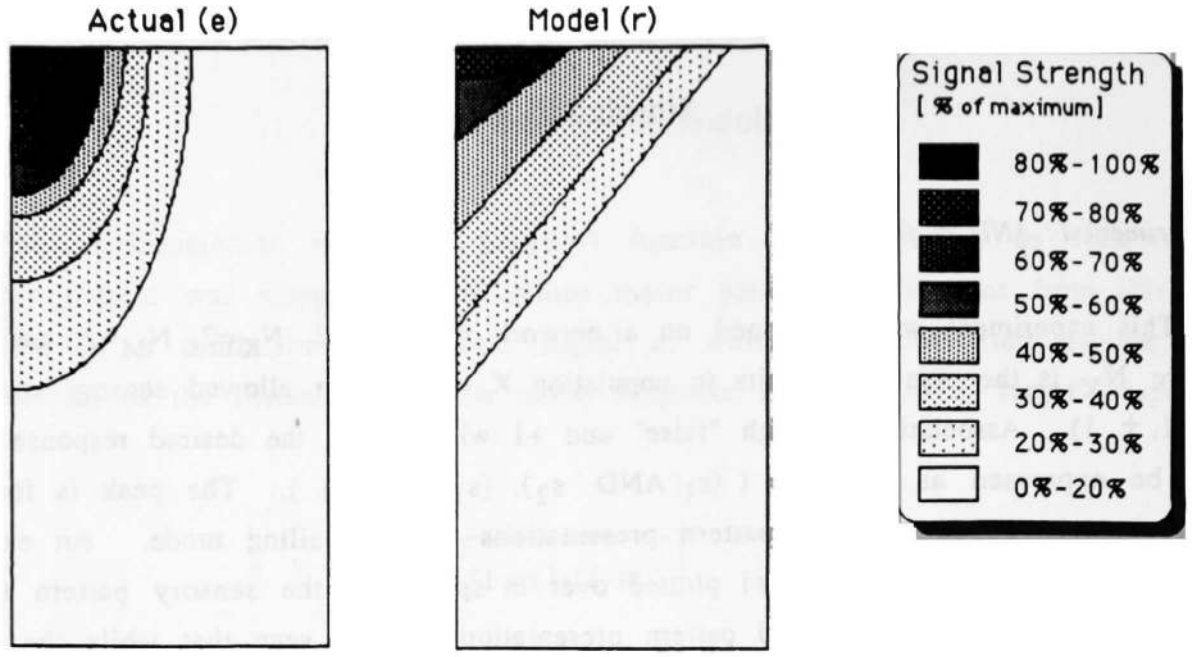
Experiment 1: AND & OR.

This experiment was performed on a network with $N_S=2$, $N_K=2$, $N_M=2$, and $N_H=12$, where N_X is the number of units in population **X**. The four allowed sensory stimuli were ($\pm$ 1, $\pm$ 1). Associating -1 with "false" and +1 with "true", the desired response pattern **t** can be expressed as (t_1,t_2) = ((s_1 AND s_2), (s_1 OR s_2)). The peak is found after approximately 50000 - 60000 pattern presentations in the flailing mode. An example of the functions **e**(**m**,**s**) and **r**(**m**,**s**) plotted over **m**-space for the sensory pattern (-1,-1) is shown in Figure 2 after 100000 pattern presentations. It is seen that while the details of the function do not seemed well learned, the peak (**t**) has been identified.

LOWNET learns to generate high values for r quite quickly, but due to its imperfect predictive power, high values of e do not come so quickly (Table 3). After 20000 pattern presentations, r is already quite high [note: the evaluation function was restricted to the range $0 < e < 0.8$]; but even at t=100000, e is not so high.

TABLE 3. Reward Maximization by LOWNET in Experiment 1.

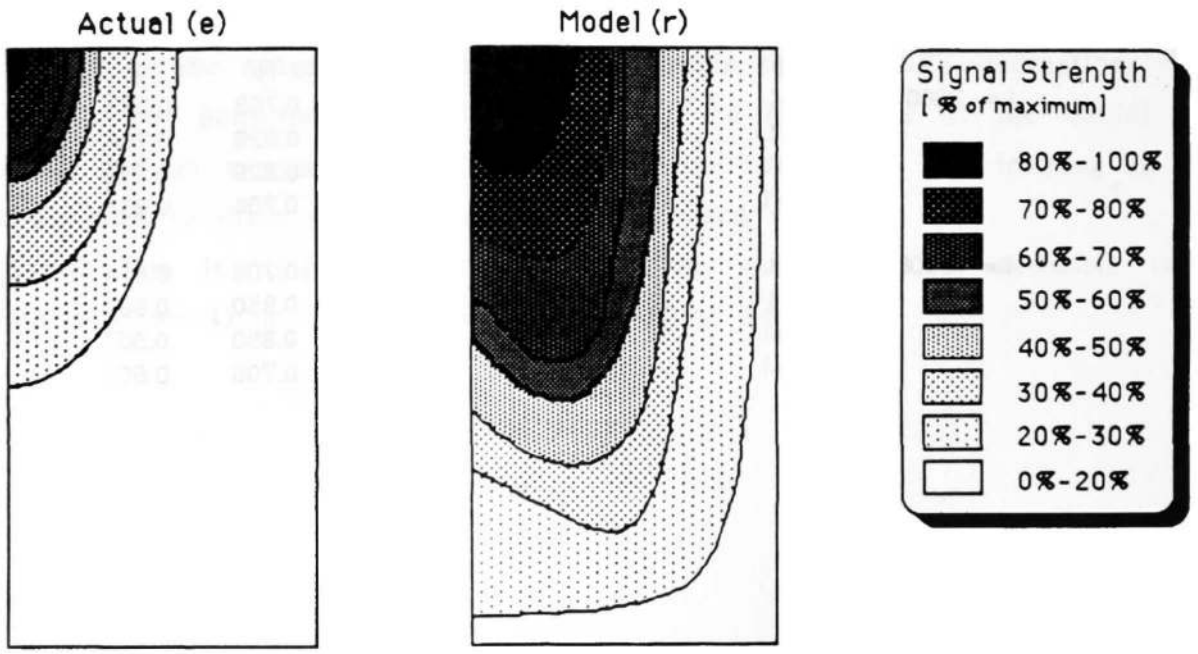
	IN 1	IN 2	OUT 1	OUT 2	R	E
t = 20000	-1	-1	-0.990	-0.842	0.703	0.583
	-1	+1	-0.596	0.620	0.829	0.264
	+1	-1	-0.597	0.619	0.829	0.264
	+1	+1	0.882	0.992	0.706	0.631
t = 100000	-1	-1	-0.999	-0.868	0.706	0.614
	-1	+1	-0.825	0.853	0.850	0.507
	+1	-1	-0.827	0.852	0.850	0.507
	+1	+1	0.897	0.999	0.708	0.651



Actual and predicted rewards as a function of the motor response.

Figure 2 (above) Contours for which the actual (left) and predicted reward values are shown for Experiment 1.

Figure 3(below) Contours for which the actual (left) and predicted reward values are shown for Experiment 2.



Experiment 2: NEGATION.

In the second experiment, a third "negation" unit was added to the set **S**. The number of possible sensory patterns was thus doubled: $\mathbf{s} = (\pm 1, \pm 1, \pm 1)$. For the case $s_3 = -1$, the target responses depended upon s_1 and s_2 just as in the first experiment; however for the case $s_3 = +1$, the signs of the target response values were inverted. In this experiment, more flailing time is required to learn to predict the evaluation signal -- approximately 150000 trials. The functions $e(\mathbf{m},\mathbf{s})$ and $r(\mathbf{m},\mathbf{s})$ are plotted over $\mathbf{m}$-space for the sensory pattern (-1,-1,-1) in Figure 3. The reward maximization carried out in LOWNET learns to generate good r-values and e-values, but does so quite slowly (300000 pattern presentations) and has real trouble with the sensory pattern (-1,-1,-1).

TABLE 4. Reward Maximization by LOWNET in Experiment 2.

	IN 1	IN 2	IN 3	OUT 1	OUT 2	R	E
t = 50000	-1	-1	-1	-0.292	0.992	0.061	0.012
	-1	-1	+1	0.628	0.725	0.852	0.317
	-1	+1	-1	-0.897	0.992	0.780	0.651
	-1	+1	+1	0.843	-0.831	0.781	0.504
	+1	-1	-1	-0.900	0.993	0.780	0.654
	+1	-1	+1	0.839	-0.824	0.782	0.496
	+1	+1	-1	0.135	0.308	0.775	0.087
	+1	+1	+1	-0.518	-0.758	0.604	0.272
t = 300000	-1	-1	-1	-0.896	0.999	0.066	0.015
	-1	-1	+1	0.982	0.939	0.896	0.704
	-1	+1	-1	-0.976	0.998	0.790	0.762
	-1	+1	+1	0.994	-0.702	0.815	0.441
	+1	-1	-1	-0.976	0.998	0.790	0.763
	+1	-1	+1	0.994	-0.694	0.818	0.433
	+1	+1	-1	0.838	0.547	0.873	0.305
	+1	+1	+1	-0.969	-0.962	0.702	0.726

Discussion

The question of how instructional information is incorporated by a learning procedure appears in various incarnations at all levels of cognitive science, from expert systems to neurophysiology to network models. The recent design/discovery of the back-propagation learning procedure addresses part of this question. While there are important issues that need to be ironed out such as biological plausibility (information passes backwards across synapses!) and psychological plausibility (too much detail in the supervisory signal!), the algorithm is so powerful and elegant that it is too attractive to dismiss. Hence it seems worthwhile to try to make it fit nicely with biological and behavioral phenomenology.

The scheme described in this paper combines two forms of back-propagation learning into an architecture that requires only a single (scalar) reward value and not the explicit training information required from the environment back-propagation learning procedure by the "standard" back-propagation algorithm.

Bibliography

1. Ackley, D., Hinton, G., and Sejnowski, T. (1985) A learning algorithm for Boltzman Machines. *Cognitive Science* **9**:147-169.

2. Rumelhart, D. E., Hinton, G., and Williams, R. W. (1986) Learning internal representations by error propagation. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. vol 1.* D. E. Rumelhart and J. L. McClelland eds. MIT/Bradford